Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehrstuhl C für Mathematik
Prof. Dr. Sjoerd Dirksen

Seminar Report

Random Sketches for Kernel Regression

Tobias Pohlen
Matriculation Number: 308743

May 2015

Contents

1 Introduction

Collecting large amounts of data has become increasingly easier in recent years. Everything from customer behaviour to stock market data can be stored efficiently due to advances in database technology for big data. In machine learning we want to use this vast amount of data in order to predict unknown quantities. One of the most common machine learning task is the problem of *regression*. It is the task of predicting a single real-valued *target* $y \in \mathbb{R}$ based on an *input* $\mathbf{x} \in \mathcal{X}$ where $\mathcal{X}$ is some non-empty set called the *domain*.

Popular applications of regression can be found in a wide variety of fields. For example, suppose one wants to predict the number of sales in the near future. In this case, the domain $\mathcal{X}$ is $\mathbb{R}$ and an input $\mathbf{x}$ encodes a certain point in time. A more sophisticated example could be an email client that wants to sort incoming emails based on their relevance to the user. When only considering the actual textual content of the email (*i.e.*, disregarding additional properties such as the sender or attachments), an email corresponds to a finite sequence of words over a dictionary $\mathcal{D}$. Here, the domain $\mathcal{X}$ is the set of all finite word sequences over $\mathcal{D}$. As a final example let us suppose one wants to predict the price of a certain asset based on recent market developments. If the state of the market at time t can be encoded by a vector $\mathbf{x}_t \in \mathbb{R}^d$, then the recent market development is a time series $\{\mathbf{x}_t : t \in T\} \subset \mathbb{R}^d$ indexed by a set T. Hence, the domain $\mathcal{X}$ is the set of all time series indexed by T.

In the commonly used *Gaussian noise model*, we are given a set of N independent training examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathbb{R}$ and we assume that the targets y_n depend on the inputs $\mathbf{x}_n$ through

$$y_n = f^*(\mathbf{x}_n) + w_n$$

for $n = 1, ..., N$ where $w_1, ..., w_N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$. We call $\sigma > 0$ the *noise level*. In this model, $f^* : \mathcal{X} \mapsto \mathbb{R}$ is the unknown *regression function* that we wish to learn from the given examples. The framework of *kernel methods* allows us to learn f^* for arbitrary domains $\mathcal{X}$ by using a suitable *kernel*. This leads to the assumption of f^* belonging to some *reproducing kernel Hilbert space (RKHS)*. We give an introduction to kernel methods in section 2.

While current database storage methods scale well with the number of training samples stored, many learning algorithms do not. In section 3 we will see that learning f^* under the Gaussian noise model paired with the assumption that f^* belongs to an RKHS reduces to solving a least-squares system with N unknowns. This learning procedure is commonly referred to as *kernel ridge regression (KRR)*. The time to solve the least-squares system exactly using standard numerical algebra techniques scales as $O(N^3)$. This cubic time complexity renders the standard approach infeasible for the kind of large problem sizes that we frequently face in big data scenarios. Although we could apply some iterative approximation scheme to speed up learning, we would like to solve the problem exactly. This is due to the fact that we can give some theoretical performance guarantees in this case. For this reason, we are not interested in approximate solutions. In an attempt to combine theoretical performance guarantees with fast learning, Yang *et al.* proposed a new learning procedure based on *random sketches* [YPW]. Their idea is to reduce the number of unknowns from N to m with $m \ll N$ while still maintaining theoretical performance guarantees. We will present and discuss their main result in section 4.

Yang *et al.* illustrated their theoretical predictions by simulating several regression problems for different kinds of random sketches. These experiments were conducted in order to illustrate the validity of their derived bounds numerically. However, the obtained results do not allow for a proper judgment of the performance on real-world data. Therefore, we perform numerical experiments in order to evaluate the performance of the approach on real-world data. In addition, we investigate the behaviour of the sparse Jonson-Lindenstrauss transform as a random sketch. We present our findings in section 5.

2 Kernel Methods

We formulated the regression problem in terms of an arbitrary domain $\mathcal{X}$. Intuitively speaking, we want to predict the target y for an input $\mathbf{x} \in \mathcal{X}$ such that the pair $(\mathbf{x}, y)$ is somewhat *similar* to the given training examples. Hence, we need some similarity measure that is large if the pairs $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ are similar. While it is easy to define a similarity measure for the real valued targets (y, y'), it can be a challenging task for the inputs $(\mathbf{x}, \mathbf{x}')$. The idea behind kernel methods is to find an embedding Φ of the domain $\mathcal{X}$ into some inner product space $\mathcal{H}$. Given such an embedding, we can measure the similarity of two inputs $\mathbf{x}, \mathbf{x}'$ using the inner product in $\mathcal{H}$. This results in the following definition of a *kernel*.

Definition 1 (Kernel)**.** *Let $\mathcal{H}$ be an inner product space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ called the* feature space*. Further, let $\Phi : \mathcal{X} \mapsto \mathcal{H}$ be a* feature map*. Then, the function*

$$k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}, (\mathbf{x}, \mathbf{x}') \mapsto \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle_{\mathcal{H}}$$

is called kernel*.*

If we can formulate a learning algorithm that depends on the training examples only through a kernel, then the learning algorithm can, in principle, be applied to arbitrary problem domains. We only need to design a kernel for the specific problem domain. The following example illustrates the use of a feature map for constructing a kernel that measures the similarity of two emails over a common dictionary.

Example 1 (Email relevance)**.** *Let us reconsider introductory example of an email client that wants to predict the relevance of an incoming email. Let $\mathcal{D} = \{w_1, ..., w_D\}$ be a dictionary of D words and let $\mathbf{x} = (w_{x_1}, ..., w_{x_K}) \in \mathcal{D}^K$ be an email of length K. We can define a feature map into the feature space $\mathbb{R}^D$ equipped with the standard dot-product as*

$$\Phi : \mathcal{X} \mapsto \mathcal{H} = \mathbb{R}^D, \mathbf{x} \mapsto (\mathbb{I}(\exists k \in [K] : w_{x_k} = w_d))_{d=1}^D$$

where $\mathbb{I}(A)$ is the indicator function that is 1 if the argument A is true and 0 otherwise. Using this feature map, the similarity of two emails $k(\mathbf{x}, \mathbf{x}')$ is the total number of words they have in common.

Obviously, we need to define a feature map Φ if we want to measure similarities for domains $\mathcal{X}$ that are not equipped with an inner product. However, using a feature map can be beneficial even if the domain is already an inner product space. The key idea is to map the domain into some higher dimensional feature space where the inner product where the inner product is a semantically more suitable similarity measure. The following example illustrates this idea for a second order polynomial feature map.

Example 2 (Polynomial feature map)**.** *Suppose our domain $\mathcal{X} = \mathbb{F}_2^D = \{0, 1\}^D$ is the set of D dimensional binary vectors. If we simply use the identity to map $\mathcal{X}$ into $\mathbb{R}^D$ endowed with the standard dot-product, then the resulting similarity measurement does not take into account possibles correlations between the entries of $\mathbf{x}$. We can incorporate such correlations into a kernel by taking the products of all combinations of two entries. This results in the* second order polynomial feature map

$$\Phi : \mathcal{X} \mapsto \mathcal{H} = \mathbb{R}^{\frac{D(D+1)}{2}}, \mathbf{x} \mapsto [x_1 x_1 \mid 2x_1 x_D \mid \ldots \mid x_2 x_2 \mid \ldots \mid 2x_2 x_D \mid \ldots \mid x_D x_D]^T.$$

Because the entries $x_d, d \in [D]$, are binary variables, each product corresponds to a logical AND *operation. If we use the the standard dot product of $\mathcal{H}$, then the similarity measure $k(\mathbf{x}, \mathbf{x}')$ also captures the similarity in terms of correlated entries. Thus, using a non-trivial feature map can result in a semantically more suitable similarity measure.*

While we only considered second order polynomials in the previous example, it seems tempting to simply increase the degree of the polynomials in order to extend the expressibility of the kernel even further. However, for a domain of dimensionality D and a polynomial of degree d, there are $D_{\mathcal{H}} := \begin{pmatrix} D+d-1 \\ d \end{pmatrix}$ possible monomials and therefore the dimensionality of the feature space is also $D_{\mathcal{H}}$. Because of the

exponential growth of $D_{\mathcal{H}}$ in d, the feature map in its given form is only computationally feasible for small values of d. However, by simple algebraic manipulations we see

$$k(\mathbf{x},\mathbf{x}') = \sum_{i,j\in[2]:i+j=2}^{\frac{D(D+1)}{2}} x_i x_j x_i' x_j' = \sum_{i=1}^{D} x_i^2 (x_i')^2 + 2\sum_{i=1}^{D}\sum_{j=1}^{D} x_i x_j x_i' x_j' = \left(\sum_{i=1}^{D} x_i x_i'\right)^2 = (\mathbf{x}\cdot\mathbf{x}')^2.$$

Hence, the resulting kernel can be computed very efficiently without computing the feature map explicitly (in time $O(D)$) by simply squaring the dot product of the input vectors. This behaviour generalizes naturally to the case where $d \geq 2$.

The fact that we might be able to evaluate a kernel efficiently without computing the feature map raises the question which functions $k : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ allow for a representation in terms of an inner product under a feature map. This leads us to the definition of *positive definite kernels* and the construction of the *reproducing kernel Hilbert space (RKHS)*.

2.1 Positive Definite Kernels

We start with some definitions regarding positive definite kernels.

Definition 2 (Kernel matrix)**.** *Let $k : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ and let $\mathbf{x}_1,...,\mathbf{x}_N \in \mathcal{X}$. The matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ with entries*

$$K_{n,m} = k(\mathbf{x}_n,\mathbf{x}_m)$$

is called kernel matrix *of k with respect to $\mathbf{x}_1,...,\mathbf{x}_N$.*

Definition 3 (Positive semi-definite matrix)**.** *A symmetric matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ is* positive semi-definite *iff*

$$\mathbf{a}^T\mathbf{K}\mathbf{a} \geq 0, \quad \textit{for all } \mathbf{a} \in \mathbb{R}^N. \tag{1}$$

Definition 4 (Positive definite kernel)**.** *A symmetric function $k : \mathcal{X}\times\mathcal{X} \mapsto \mathbb{R}$ is called* positive definite kernel *on $\mathcal{X}$ iff for all $N \in \mathbb{N}$ and all $\mathbf{x}_1,...,\mathbf{x}_N \in \mathcal{X}$ the kernel matrix $\mathbf{K}$ of k with respect to $\mathbf{x}_1,...,\mathbf{x}_N$ is positive semi-definite.*

Note that there is a naming discrepancy between the literature on kernels and the linear algebra literature. While one calls matrices for which (4) holds positive *semi*-definite, one calls the corresponding kernels positive *definite*.

The following theorem answers the pending question which function s admit a representation in terms of an inner product of feature maps.

Theorem 1. *The class of kernels (Definition 1) is identical to the class of positive definite kernels (Definition 4).*

In order to prove the theorem, we need the following well-known inequality from linear algebra for positive definite kernels.

Lemma 1 (Cauchy-Schwarz inequality for kernels)**.** *Let k be a positive definite kernel and let $\mathbf{x}_1,\mathbf{x}_2 \in \mathcal{X}$. Then, it holds*

$$|k(\mathbf{x}_1,\mathbf{x}_2)|^2 \leq k(\mathbf{x}_1,\mathbf{x}_1)k(\mathbf{x}_2,\mathbf{x}_2).$$

Proof (Lemma 1). Consider the 2×2 positive semi-definite kernel matrix $\mathbf{K}$ with entries $K_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j)$ for $i,j = 1,2$. The positive semi-definiteness implies that all eigenvalues of $\mathbf{K}$ are non-negative. Using that the product of the eigenvalues equals the determinant of $\mathbf{K}$ we get

$$0 \leq \det\mathbf{K} = K_{1,1}K_{2,2} - |K_{1,2}|^2 = k(\mathbf{x}_1,\mathbf{x}_1)k(\mathbf{x}_2,\mathbf{x}_2) - |k(\mathbf{x}_1,\mathbf{x}_2)|^2.$$

By rearrangement we get the desired inequality. □

Proof (Theorem 1). We first show that all kernels are also positive definite. Let $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ be a kernel with feature map $\Phi : \mathcal{X} \mapsto \mathcal{H}$. We have to show that the kernel matrix $\mathbf{K}$ is positive semi-definite for all choices of $N \in \mathbb{N}$ and $\mathbf{x}_1, ..., \mathbf{x}_N \in \mathcal{X}$. Let $\mathbf{a} \in \mathbb{R}^N$. It holds

$$\begin{aligned}
\mathbf{a}^T \mathbf{K} \mathbf{a} &= \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m K_{n,m} \\
&= \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m \langle \Phi(\mathbf{x}_n), \Phi(\mathbf{x}_m) \rangle_{\mathcal{H}} \\
&= \left\langle \sum_{n=1}^{N} a_n \Phi(\mathbf{x}_n), \sum_{m=1}^{N} a_m \Phi(\mathbf{x}_m) \right\rangle_{\mathcal{H}} \\
&= \left\| \sum_{n=1}^{N} a_n \Phi(\mathbf{x}_n) \right\|_{\mathcal{H}}^2 \geq 0
\end{aligned}$$

because of the non-negativity of the norm $\| \cdot \|_{\mathcal{H}}$.

Now we prove the reverse direction by constructing a feature map Φ into an inner product space $\mathcal{H}$ from a positive definite kernel k. We choose $\mathcal{H}$ to be the following linear subspace of $\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \mapsto \mathbb{R}\}$

$$\mathcal{H} := \text{span}\{k(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$$

and we take the corresponding feature map to be

$$\Phi : \mathcal{X} \mapsto \mathcal{H}, \mathbf{x} \mapsto k(\cdot, \mathbf{x}).$$

Φ maps an input point $\mathbf{x}$ to a function that measures the similarity of $\mathbf{x}$ to all other input points. In order for Φ to be a feature map according to definition 1, we need to define an inner product on $\mathcal{H}$. Let $f = \sum_{i=1}^{I} \alpha_i k(\cdot, \mathbf{x}_i), g = \sum_{j=1}^{J} \beta_j k(\cdot, \mathbf{x}'_j) \in \mathcal{H}$. We define their inner product as

$$\langle f, g \rangle_{\mathcal{H}} = \left\langle \sum_{i=1}^{I} \alpha_i k(\cdot, \mathbf{x}_i), \sum_{j=1}^{J} \beta_j k(\cdot, \mathbf{x}'_j) \right\rangle_{\mathcal{H}} := \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}'_j).$$

Although the coefficients α_i of f and β_j of g might not be unique, the inner product is well defined as we can easily see

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}'_j) \overset{*}{=} \sum_{j=1}^{J} \beta_j \underbrace{\sum_{i=1}^{I} \alpha_i k(\mathbf{x}'_j, \mathbf{x}_i)}_{= f(\mathbf{x}'_j)} = \sum_{j=1}^{J} \beta_j f(\mathbf{x}'_j) = \sum_{i=1}^{I} \alpha_i g(\mathbf{x}_i),$$

where we used the symmetry of k in $*$. This shows that the inner product does not depend on the particular choice of coefficients $\{\alpha_i\}_{i=1}^{I}$ and $\{\beta_j\}_{j=1}^{J}$, respectively. Furthermore, we see that the symmetry of k also implies the symmetry of $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. Using the same manipulations we can prove that the inner product is bilinear. Let $h = \sum_{k=1}^{K} \gamma_k k(\cdot, \mathbf{x}''_k)$. Then,

$$\langle f + h, g \rangle_{\mathcal{H}} = \sum_{i=1}^{I} \alpha_i g(\mathbf{x}_i) + \sum_{k=1}^{K} \gamma_k g(\mathbf{x}''_k) = \langle f, g \rangle_{\mathcal{H}} + \langle h, g \rangle_{\mathcal{H}}.$$

Now, we have to prove that $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is positive definite and $\langle f, f \rangle_{\mathcal{H}} = 0$ if and only if $f = 0$. First, we note that we have

$$\langle f, f \rangle_{\mathcal{H}} = \sum_{i=1}^{I} \sum_{j=1}^{J} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

because of the positive definiteness of k. Also note that the coefficients α_i are arbitrary. Now let $f_1, ..., f_N \in \mathcal{H}$ and $a_1, ..., a_N \in \mathbb{R}$. We get

$$\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m \langle f_n, f_m\rangle_{\mathcal{H}} = \left\langle \sum_{n=1}^{N} a_n f_n, \sum_{n=1}^{N} a_n f_n \right\rangle_{\mathcal{H}} = \langle \tilde{f}, \tilde{f}\rangle_{\mathcal{H}} \geq 0.$$

Hence, the inner product is positive definite. For the final step, we use the *reproducing property*

$$\langle k(\cdot, \mathbf{x}), f\rangle_{\mathcal{H}} = \sum_{i=1}^{I} \alpha_i k(\mathbf{x}, \mathbf{x}_i) = f(\mathbf{x}) \tag{2}$$

for $\mathbf{x} \in \mathcal{X}$. Using lemma 1 we get

$$|f(\mathbf{x})|^2 = |\langle k(\cdot, \mathbf{x}), f\rangle_{\mathcal{H}}|^2 \leq \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x})\rangle_{\mathcal{H}} \langle f, f\rangle_{\mathcal{H}}.$$

Thus $\langle f, f\rangle_{\mathcal{H}} = 0$ implies $|f(\mathbf{x})|^2 = 0$ for all $\mathbf{x} \in \mathcal{X}$. Therefore, it implies $f = 0 \in \mathcal{H}$. Finally, we have to show that $k(\mathbf{x}, \mathbf{x}')$ corresponds to the inner product of the feature vectors in $\mathcal{H}$. It holds

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}')\rangle_{\mathcal{H}} = \langle k(\cdot, \mathbf{x}), k(\cdot, \mathbf{x}')\rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}').$$

□

Theorem 1 shows that there are two ways to construct kernels: we can either find an explicit feature map into some inner product space or we can state some function and prove that it is a positive definite kernel.

By definition, the space $\mathcal{H}$ that we constructed in the above proof is a pre-Hilbert space (a vector space with an inner product). By completing $\mathcal{H}$ in the norm $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$ we obtain a *reproducing kernel Hilbert space (RKHS)*.

Definition 5 (Reproducing kernel hilbert space)**.** *Let $\mathcal{X}$ be a non-empty set. A space $\mathcal{H} \subset \mathbb{R}^{\mathcal{X}}$ with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is called a* reproducing kernel Hilbert space *iff there exists a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ such that the following properties hold:*

1. *$\langle f, k(\cdot, \mathbf{x})\rangle_{\mathcal{H}} = f(\mathbf{x})$ for all $f \in \mathcal{H}$ and $\mathbf{x} \in \mathcal{X}$;*
2. *$\mathcal{H}$ is the completion of span$\{k(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}$,* i.e.*, $\mathcal{H} = \overline{span\{k(\cdot, \mathbf{x}) : \mathbf{x} \in \mathcal{X}\}}$.*

In the following we will refer to the completed space $\mathcal{H}$ as the RKHS of a kernel k.

2.2 The Representer Theorem

In machine learning we generally assume that there exists an *unknown* distribution $P(\mathbf{x}, y)$ on $\mathcal{X} \times \mathbb{R}$ from which all data is generated. If we factorize the joint distribution as

$$P(\mathbf{x}, y) = P(y \mid \mathbf{x})P(\mathbf{x})$$

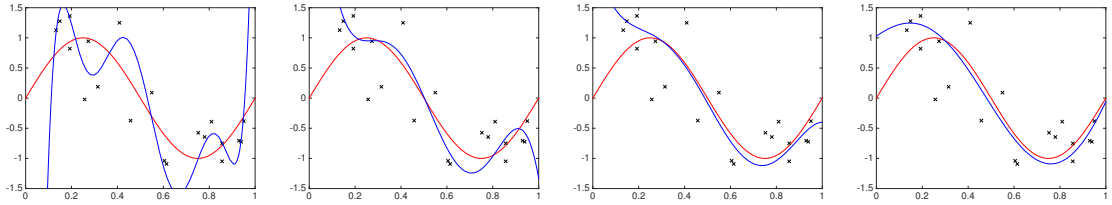
then we see that there are two defining factors. The distribution $P(\mathbf{x})$ defines the probability of generating a certain input $\mathbf{x}$ and the conditional distribution $P(y \mid \mathbf{x})$ determines targets for a given input $\mathbf{x}$. In the Gaussian noise model, we assume that $P(y \mid \mathbf{x})$ is a Gaussian distribution with mean $f^*(\mathbf{x})$ and variance σ^2. *Learning* a function $f^* : \mathcal{X} \mapsto \mathbb{R}$ refers to the task of finding a minimizer of the *expected risk functional*

$$R(f) = \int_{\mathcal{X}\times\mathbb{R}} \mathcal{L}(y, f(\mathbf{x}))\mathrm{d}P(\mathbf{x}, y)$$

where $\mathcal{L} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+ \cup \{\infty\}$ is a *loss function* with $\mathcal{L}(y, y) = 0$. $\mathcal{L}(y, f(\mathbf{x}))$ is the loss incurred from predicting the value $f(\mathbf{x})$ for the data point $\mathbf{x}$ if the correct value is y.

Because we cannot minimize the expected risk $R(f)$ directly since the distribution $P(\mathbf{x}, y)$ is unknown in general, intuition suggests that we can instead minimize the *empirical risk* if the training examples are generated i.i.d. from $P(\mathbf{x}, y)$

$$R_{emp}(f) = \frac{1}{N}\sum_{n=1}^{N} \mathcal{L}(y_n, f(\mathbf{x}_n)).$$



$\lambda = 0, R_{emp} = 0.1893$ $\lambda = 10^{-8}, R_{emp} = 0.3040$ $\lambda = 10^{-7}, R_{emp} = 0.3362$ $\lambda = 10^{-6}, R_{emp} = 0.3566$

Figure 1: The red curve shows the regression function f^*. The blue curve shows the estimator that is learned from the given data points (black) by minimizing the regularized risk R_{reg} with $\Omega(f) = \lambda \|f\|^2_{\mathcal{H}}$. We make two important observations: First, the larger the regularization parameter λ is, the smoother the learned estimator becomes. Second, although the empirical risk increases for increasing values of λ, the expected risk decreases.

However, minimizing the empirical risk is an ill-posed problem as there can be many functions that achieve minimum risk. For this reason, one usually learns a function by minimizing a *regularized risk functional* of the form

$$R_{reg}(f) = R_{emp}(f) + \lambda \Omega(f), \quad \lambda > 0,$$

where Ω is a *regularization functional* that is small if f is smooth. Figure 1 illustrates the concept. By minimizing the regularized risk functional we attempt to find a trade-off between minimum empirical risk and the smoothness of f. If we minimize R_{ref} over some RKHS, then Ω usually is a function of the squared norm $\|f\|^2_{\mathcal{H}}$ of f. The following theorem states that the solution $\hat{f}$ to the problem

$$\min_{f \in \mathcal{H}} R_{reg}(f)$$

always admits an easily computable form.

Theorem 2 (Representer theorem [SS01, Theorem 4.2])**.** *Let $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathbb{R}$. Furthermore, let $\Omega : \mathbb{R}^+ \mapsto \mathbb{R}^+$ be a strictly increasing function and let $\mathcal{L} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^+ \cup \{\infty\}$ be an arbitrary loss function. Then, each minimizer $\hat{f} \in \mathcal{H}$ of the regularized risk*

$$R_{reg}(f) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}(y_n, f(\mathbf{x}_n)) + \Omega(\|f\|^2_{\mathcal{H}}) \tag{3}$$

admits a representation of the form

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}, \mathbf{x}_n)$$

where $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$.

Before we prove theorem 2, let us emphasize its importance. The RKHS of a kernel k can be an infinite dimensional vector space. Thus, there could be infinitely many parameters to optimize. The theorem states, however, that if we minimize a regularized risk functional, then all minimizers of the risk can be expressed as a linear combination of the feature maps $k(\cdot, \mathbf{x}_n)$. Therefore, instead of optimizing over possibly infinitely many parameters, we only have to optimize over the N weighting coefficients α_n.

Proof. Proof (Theorem 2). Let $\hat{f} \in \mathcal{H}$ be a minimizer of the regularized risk functional (3) and consider a decomposition of the form

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}, \mathbf{x}_n) + \hat{f}_{\perp}(\mathbf{x})$$

where $\alpha_1, ..., \alpha_N \in \mathbb{R}$ and $\langle k(\cdot, \mathbf{x}_n), \hat{f}_\perp \rangle_{\mathcal{H}} = 0$ for $n \in [N]$. By the reproducing property (2) of the RKHS, we can write $\hat{f}(\mathbf{x}_n)$ as

$$\begin{aligned}\hat{f}(\mathbf{x}_n) = \langle k(\cdot, \mathbf{x}_n), \hat{f} \rangle_{\mathcal{H}} &= \left\langle k(\cdot, \mathbf{x}_n), \sum_{m=1}^{N} \alpha_m k(\cdot, \mathbf{x}_m) + \hat{f}_\perp \right\rangle_{\mathcal{H}} \\ &= \sum_{m=1}^{N} \alpha_m \langle k(\cdot, \mathbf{x}_n), k(\cdot, \mathbf{x}_m) \rangle_{\mathcal{H}} + \underbrace{\langle k(\cdot, \mathbf{x}_n), \hat{f}_\perp \rangle_{\mathcal{H}}}_{=0} = \sum_{m=1}^{N} \alpha_m k(\mathbf{x}_n, \mathbf{x}_m)\end{aligned}$$

Therefore, the value of the empirical risk R_{emp} does not depend on the orthogonal complement $\hat{f}_\perp$. Furthermore, it holds

$$\Omega(\|\hat{f}\|_{\mathcal{H}}^2) = \Omega\left(\left\|\sum_{n=1}^{N} \alpha_n k(\cdot, \mathbf{x}_n)\right\|_{\mathcal{H}}^2 + \|\hat{f}_\perp\|_{\mathcal{H}}^2\right) \geq \Omega\left(\left\|\sum_{n=1}^{N} \alpha_n k(\cdot, \mathbf{x}_n)\right\|_{\mathcal{H}}^2\right)$$

due to the monotonicity of Ω. This implies that the regularizer is minimized for $\hat{f}_\perp = 0$. Combining both results yields that $\hat{f}$ can be written as

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n k(\mathbf{x}, \mathbf{x}_n)$$

□

3 Kernel Ridge Regression (KRR)

Let us now consider the regression problem where we have N i.i.d. training examples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N) \in \mathcal{X} \times \mathbb{R}$ that are generated from the Gaussian noise model

$$y_n = f^*(\mathbf{x}_n) + w_n \tag{4}$$

where $w_1, ..., w_N \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2)$ and $\sigma > 0$. Our goal is to learn the unknown regression function $f^* : \mathcal{X} \mapsto \mathbb{R}$. Given the Gaussian nature of the targets y_n, it is naturally to approach the problem form a statistical point of view. The relationship (4) is equivalent to the assumption that the targets are distributed as $y_n \sim \mathcal{N}(f^*(\mathbf{x}_n), \sigma^2)$. Hence, the regression function f^* is the mean of a Gaussian distribution. We can use the framework of *Maximum Likelihood Estimation (MLE)* in order derive a learning criterion. The goal of MLE is to maximize the joint likelihood $p(y_1, ..., y_N \mid f^*(\mathbf{x}_1), ..., f^*(\mathbf{x}_N))$ with respect to the regression function f^*. Because the targets are independent, we can factorize the joint likelihood as

$$p(y_1, ..., y_N \mid f^*(\mathbf{x}_1), ..., f^*(\mathbf{x}_N)) = \prod_{n=1}^{N} p(y_n \mid f^*(\mathbf{x}_n)). \tag{5}$$

The maximizer of (5) is invariant under a transformation with a monotonically increasing function. Thus, maximizing (5) is equivalent to maximizing the Log-Likelihood function that is obtained from (5) by applying the logarithm

$$\begin{aligned} L(f^*) = \log \prod_{n=1}^{N} p(y_n \mid f^*(\mathbf{x}_n)) &= \sum_{n=1}^{N} \log \mathcal{N}(y_n \mid f^*(\mathbf{x}_n), \sigma^2) \\ &= \sum_{n=1}^{N} \left(\log \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} (f^*(\mathbf{x}_n) - y_n)^2 \right). \end{aligned} \tag{6}$$

Note that the term $\frac{1}{\sqrt{2\pi\sigma^2}}$ does not depend on f^* and can thus be omitted from the optimization problem. Furthermore, scaling with a positive constant does not change the maximizing argument. Therefore, maximizing (6) is equivalent to maximizing

$$-\frac{1}{N} \sum_{n=1}^{N} (f^*(\mathbf{x}_n) - y_n)^2.$$

Finally, we note that instead of maximizing a function we can also minimize its negative. This results in the following learning criterion for the maximum likelihood estimation of f^*

$$\min_f \frac{1}{N} \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n)^2.$$

We see that determining f^* in the maximum likelihood sense results in the same optimization problem as finding f^* by minimizing the empirical risk when using the squared loss function $\mathcal{L}_{squared}(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$. We have seen in section 2.2 that while minimizing the empirical risk seems intuitive, we need a regularizer Ω in order to obtain a well-posed optimization problem. If we assume f^* to belong to a RKHS $\mathcal{H}$ for some kernel k, then the regularized optimization problem can be stated as

$$\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n)^2 + \lambda \|f\|_{\mathcal{H}}^2 \tag{P_p}$$

where $\lambda > 0$ is the regularization parameter. It controls the trade-off between the smoothness of the solution and the fit to the training examples.

3.1 Dual Formulation

In this section, we derive the so called *dual form* of the optimization problem (P_p). This form allows us to learn the function f^* in practice using standard methods from numerical linear algebra. We start by noting

that the regularized risk functional in (P_p) fulfills the requirements of the representer theorem (Theorem 2) as long as $\lambda > 0$. Therefore, any minimum $\hat{f}$ of the optimization problem can be expressed in terms of a linear combination of the feature maps. Thus, optimizing (P_p) over the possibly infinite dimensional Hilbert space $\mathcal{H}$ is equivalent to optimizing the following problem over the space $\mathbb{R}^N$. Applying the representer theorem yields

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)-y_n\right)^2+\lambda\left\langle\sum_{n=1}^{N}\alpha_n k(\cdot,\mathbf{x}_n),\sum_{n=1}^{N}\alpha_n k(\cdot,\mathbf{x}_n)\right\rangle_{\mathcal{H}}.$$

Let $\mathbf{K}$ be the kernel matrix of k with respect to the training examples $\mathbf{x}_1,\ldots,\mathbf{x}_N$. Using the bi-linearity of the inner product, we can simplify the objective function further

$$\begin{aligned}
&\frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)-y_n\right)^2+\lambda\left\langle\sum_{n=1}^{N}\alpha_n k(\cdot,\mathbf{x}_n),\sum_{n=1}^{N}\alpha_n k(\cdot,\mathbf{x}_n)\right\rangle_{\mathcal{H}}\\
=&\frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)-y_n\right)^2+\lambda\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m\langle k(\cdot,\mathbf{x}_n),k(\cdot,\mathbf{x}_m)\rangle_{\mathcal{H}}\\
\overset{*}{=}&\frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)-y_n\right)^2+\lambda\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m k(\mathbf{x}_m,\mathbf{x}_n)\\
=&\frac{1}{N}\sum_{n=1}^{N}\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)-y_n\right)^2+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}. \qquad (7)
\end{aligned}$$

where we used the reproducing property of the inner product for $*$. Now let $\mathbf{y}=[y_1 \mid \ldots \mid y_N]^T\in\mathbb{R}^N$. Expanding the quadratic term yields

$$\begin{aligned}
&\frac{1}{N}\sum_{n=1}^{N}\left[\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)\right)^2-2\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)y_n+y_n^2\right]+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\\
=&\frac{1}{N}\sum_{n=1}^{N}\underbrace{\left(\sum_{m=1}^{N}\alpha_m k(\mathbf{x}_n,\mathbf{x}_m)\right)^2}_{=(\mathbf{K}\boldsymbol{\alpha})_n^2}-\frac{2}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{y}+\frac{1}{N}\|\mathbf{y}\|_2^2+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\\
=&\frac{1}{N}\|\mathbf{K}\boldsymbol{\alpha}\|_2^2-\frac{2}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{y}+\frac{1}{N}\|\mathbf{y}\|_2^2+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\\
=&\frac{1}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{K}\boldsymbol{\alpha}-\frac{2}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{y}+\frac{1}{N}\|\mathbf{y}\|_2^2+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}
\end{aligned}$$

where the last step follows from the symmetry of the kernel matrix $\mathbf{K}$. Note that the term $\frac{1}{N}\|\mathbf{y}\|_2^2$ does not depend on $\boldsymbol{\alpha}$ and can therefore be omitted from the optimization problem. The final *dual form* of the optimization problem is thus given by

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N}\frac{1}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{K}\boldsymbol{\alpha}-\frac{2}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{y}+\lambda\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}. \qquad (P_d)$$

Remark 1 (Regularization term)**.** *The regularizer $\|f\|_{\mathcal{H}}^2$ transforms to the term $\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}$ in the dual form of the learning problem. We will see in the next section that we have to compute the so-called* matrix square root *of $\mathbf{K}$ if we want to optimize (P_d) directly. In order to avoid the computational overhead that is associated with the computation of the matrix square root, one often drops the dependency of the regularization term on the kernel matrix $\mathbf{K}$. This results in the following optimization problem*

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N}\frac{1}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{K}\boldsymbol{\alpha}-\frac{2}{N}\boldsymbol{\alpha}^T\mathbf{K}\mathbf{y}+\lambda\boldsymbol{\alpha}^T\boldsymbol{\alpha}. \qquad (8)$$

In many machine learning books, kernel ridge regression is introduced using this approximate regularizer. Not only does this regularizer reduce computation time, it also improves the numerical conditioning of the optimization problem. We see that the problem (8) is convex in $\boldsymbol{\alpha}$. Therefore, the first-order sufficient condition is given by

$$\frac{2}{N}\mathbf{K}\mathbf{K}\boldsymbol{\alpha} - \frac{2}{N}\mathbf{K}\mathbf{y} + 2\lambda\boldsymbol{\alpha} = 0 \Leftrightarrow (\mathbf{K}\mathbf{K} + N\lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{K}\mathbf{y}.$$

Hence, learning $\boldsymbol{\alpha}$ requires us to invert the matrix $\mathbf{K}\mathbf{K} + N\lambda\mathbf{I}$. It often happens that the kernel matrix $\mathbf{K}$ is (close to) a singular matrix. In these cases, the regularization term makes sure that the matrix $\mathbf{K}\mathbf{K} + N\lambda\mathbf{I}$ is regular.

Remark 2 (Dual Form)**.** *Let $\Phi : \mathcal{X} \mapsto \mathbb{R}^D$ be the feature map that is associated with the kernel k. The optimization problem (P_d) can be seen as the dual form of the following optimization problem (See [SGV98] for further derivations)*

$$\begin{aligned} &\min_{\mathbf{w}\in\mathbb{R}^D} \ \lambda\|\mathbf{w}\|_2^2 + \sum_{n=1}^{N} \xi_n^2 \\ &\textit{subject to} \quad y_n - \mathbf{w}\cdot\Phi(\mathbf{x}_n) = \xi_n \ \textit{ for } n \in [N]. \end{aligned}$$

In this setting, we can also see that Kernel Ridge Regression performs regularized linear regression in the feature space $\mathcal{H}$. For this reason, one sometimes refers to Kernel Ridge Regression as generalized linear regression*.*

3.2 Numerical Implementation

Solving the problem (P_d) can be done using standard methods from numerical linear algebra. While we could leverage the first order sufficient condition in order to find the optimal $\boldsymbol{\alpha}$, this approach is not numerically stable. This is due to the fact that the condition number of $\mathbf{K}^2$ is the squared condition number of $\mathbf{K}$. If we used the first order necessary condition, then we would have to solve the linear system

$$(\mathbf{K}\mathbf{K} + \lambda N\mathbf{K})\boldsymbol{\alpha} = \mathbf{K}\mathbf{y}.$$

The condition of the linear system is determined by the condition number of the matrix $\mathbf{K}\mathbf{K} + \lambda N\mathbf{K}$. While the regularization term improves the numerical conditioning of the problem, we can solve it more stably using a QR-decomposition.

Recall that any symmetric positive matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ has an eigenvalue decomposition of the form

$$\mathbf{K} = V\Lambda V^T$$

where $V \in \mathbb{R}^{N\times N}$ and $\Lambda = \text{diag}(\lambda_1, ...\lambda_N)$ with $\lambda_n \geq 0$ for all $n = 1, ..., N$. We now define $\sqrt{\Lambda} := \text{diag}(\sqrt{\lambda_1}, ..., \sqrt{\lambda_N})$ which lets us define the square root of the matrix $\mathbf{K}$ as

$$\sqrt{\mathbf{K}} := \sqrt{\Lambda}V^T.$$

Using this definition of a matrix square root, we can reformulate (7) as

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}\left\| \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda N}\sqrt{\mathbf{K}} \end{bmatrix} \boldsymbol{\alpha} - \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \right\|_2^2. \tag{9}$$

We can solve the problem in this form using a QR-decomposition $\mathbf{Q}\mathbf{R} = \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda N}\sqrt{\mathbf{K}} \end{bmatrix}$ [GVL13, Chapter 5] where $\mathbf{Q} \in \mathbb{R}^{2N\times 2N}$ is an orthogonal matrix and $\mathbf{R} \in \mathbb{R}^{N\times 2N}$ is an upper triangle matrix. Because $\mathbf{Q}$ is an isometry on $(\mathbb{R}^{2N}, \|\cdot\|_2)$ and $\mathbf{Q}^T = \mathbf{Q}^{-1}$, (9) is equivalent to

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \frac{1}{N}\left\| \mathbf{R}\boldsymbol{\alpha} - \mathbf{Q}^T \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix} \right\|_2^2.$$

In this setting, we can easily solve for $\boldsymbol{\alpha}$ by backwards substitution. Note that the condition number of the QR-decomposition is the same as the condition number of the original matrix because $\mathbf{Q}$ is an isometry.

The total runtime of the optimization algorithm scales as $O(N^3)$ because the eigendecomposition of $\mathbf{K}$ as well as the QR-decomposition of the combined matrix $\begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda N}\sqrt{\mathbf{K}} \end{bmatrix}$ require $O(N^3)$ steps. While this is certainly computationally feasible for small N, the approach renders infeasible for very large N (*i.e.*, $N \gg 10,000$). In an attempt to reduce the size of the optimization problem while still maintaining certain theoretical performance guarantees, Yang *et al*. proposed to project the problem onto a smaller space using *random sketches*. We discuss their approach in the following section.

3.3 Bibliographical Notes

The two main results (Theorems 1 and 2) and their original proofs can be found in [SS01]. We adapted their formulation to the specific case of regression and included some additional explanations. Although we formulated the specific derivation of the KRR optimization problem using the representer theorem, there are numerous other sources that follow different paths. Saunders *et al*. derived a formulation of problem (P_d) using Lagrange multipliers [SGV98]. Another derivation that uses a plug-in approach can be found in chapter 6 of [Bis06]. In chapter 7.5, Murphy discusses the idea of ridge regression in a probabilistic context and shows how the squared regularizer can be interpreted as a Gaussian prior on the weight vector [Mur12]. Finally, numerical solvers for (regularized) least-squares systems can be found in chapters 5 and 6 of [GVL13].

4 Random Sketches for KRR

In the previous section, we saw that the time complexity of the KRR optimization problem (P_d) is $O(N^3)$. Furthermore, the space required in order to store the kernel matrix $\mathbf{K}$ scales as $O(N^2)$. These complexities show that KRR is certainly computationally feasible for moderately sized problems and can even be considered fast compared to other learning methods that require more sophisticated optimization methods. While we could apply an iterative approximation scheme in order to speed up training for large-scale problems, it is difficult to give theoretical performance guarantees for approximate solutions.

In an attempt to combine both, fast training and a theoretical performance guarantee, Yang *et al*. proposed to reduce the size of the optimization problem (P_d) by linearly projecting the matrices onto a lower dimensional subspace. Let $\mathbf{S} \in \mathbb{R}^{m \times N}$ be a matrix that projects $\mathbb{R}^N$ onto some $m \ll N$ dimensional linear subspace. The projected (or *sketched*) version of (P_d) is given by

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^m} \frac{1}{N} \boldsymbol{\alpha}^T \mathbf{S}\mathbf{K}\mathbf{K}\mathbf{S}^T \boldsymbol{\alpha} - \frac{2}{N} \boldsymbol{\alpha}^T \mathbf{S}\mathbf{K}\mathbf{y} + \lambda \boldsymbol{\alpha}^T \mathbf{S}\mathbf{K}\mathbf{S}^T \boldsymbol{\alpha}. \tag{$P_{\mathbf{S},d}$}$$

This is a quadratic optimization problem with m unknowns and can be solved in time $O(m^3)$ using the method discussed in section 3.2. Let $\hat{\boldsymbol{\alpha}}$ be the solution to $(P_{\mathbf{S},d})$, then the *sketched estimator* takes the form

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{N} (\mathbf{S}^T \hat{\boldsymbol{\alpha}})_n k(\mathbf{x}, \mathbf{x}_n).$$

There are many well-established methods for dimensionality reduction by linear projections (*e.g*., PCA [Bis06, Chapter 12], LDA [Bis06, Chapter 4]). However, most of these methods are computationally at least as expensive as solving the original optimization problem (P_d). For example, PCA requires the computation of a singular value decomposition of the kernel matrix. Computing a singular value decomposition takes time $O(N^3)$ (see *e.g*. [GVL13, Chapter 8]) as well. Therefore, these methods are computationally infeasible for large N, too. For this reason, Yang *et al*. proposed to use random projection matrices $\mathbf{S}$. They call these special projection matrices *random sketches*. Usually, these matrices can be sampled very efficiently in just $O(mN)$ steps. This makes random sketches an attractive choice for dimensionality reduction of large-scale problems.

4.1 Random Sketches

Random Sketches are a computationally efficient way to reduce the dimensionality of a dataset. The idea is to project the data points onto a random linear subspace. Let $\mathbf{S} \in \mathbb{R}^{m \times D}$ be a (random) projection matrix with $m \ll D$. If $\mathbf{S}$ can be sampled efficiently (*i.e*., in time $O(mD)$), then the runtime for projecting N points onto the lower dimensional subspace defined by the rowspan of $\mathbf{S}$ scales as $O(mDN)$[1]. This makes random sketches an attractive dimensionality reduction technique for large problem instances (*i.e*., if N and D are very large).

The idea of reducing the dimensionality of a dataset by projecting it onto a random lower dimensional subspace is due to Johnson and Lindenstrauss who proved the following theorem that is known as the Johnson-Lindenstrauss lemma (JL lemma) [JL84].

Theorem 3 (Johnson-Lindenstrauss Lemma (as stated in [FR13, Lemma 9.35])**.** *Let* $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^D$ *and* $\varepsilon > 0$*. If* $m > C\varepsilon^{-2} \log(N)$*, then there exists a matrix* $\mathbf{S} \in \mathbb{R}^{m \times D}$ *such that*

$$(1-\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{S}(\mathbf{x}_i - \mathbf{x}_j)\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{10}$$

for all $i, j \in [N]$*. The constant* $C > 0$ *is universal.*

The condition (10) says that there exists a sketch matrix $\mathbf{S}$ that almost acts as an isometry on the data points. The pair-wise distances are approximately being preserved while projecting the points onto the m dimensional subspace. Further, the theorem states that we can potentially achieve a considerable dimensionality reduction as the dimensionality of the subspace does not depend on the dimensionality D of the points

[1] We will see later that this runtime can be improved further by exploiting sparsity in $\mathbf{S}$.

and only depends logarithmically on the number of points N. Here, we are not directly interested in reducing the dimensionality of a given set of points but we want to reduce the size of the kernel matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ using a random sketch $\mathbf{S} \in \mathbb{R}^{m\times N}$ via $\mathbf{SKS}^T \in \mathbb{R}^{m\times m}$. Yang *et al*. proved that certain random sketches are suitable for this task with high probability. In their publication, they focus on *subgaussian* sketches, which we introduce below, and *ROS* sketches, which we do not consider in this seminar paper. In addition to subgaussian sketches, we also consider the class of *sparse Johnson-Lindenstrauss transforms* [KN14]. By actively enforcing a fixed column sparsity s into the sampling process, sparse Johnson-Lindenstrauss transforms are particularly suitable for large-scale dimensionality reduction.

Subgaussian sketches. We say that a random vector Y on $\mathbb{R}^N$ is *subgaussian* if for all $\mathbf{x} \in \mathbb{R}^N$ with $\|\mathbf{x}\|_2 = 1$ it holds

$$\mathbb{P}[|\langle Y, \mathbf{x}\rangle| \geq t] \leq \beta e^{-\kappa t^2}, \quad \text{for all } t > 0, \tag{11}$$

for some constants $\beta, \kappa > 0$. A matrix $\mathbf{S} \in \mathbb{R}^{m\times N}$ with independent mean-zero subgaussian rows is called a *subgaussian sketch*. This class includes, among others, matrices with i.i.d. standard Gaussian entries and matrices with i.i.d. Rademacher (random sign flips) entries. We will use these two particular types of subgaussian sketches for our numerical evaluation of the sketched estimator.

Sparse Johnson-Lindenstrauss Transform (SJLT). Following [BDN15], we define the sparse Johnson-Lindenstrauss Transform (SJLT) using two sets of independent random variables. Let $\{\xi_{i,j}\}_{i=1,j=1}^{m,N}$ be independent Rademacher random variables; *i.e*.,

$$\mathbb{P}[\xi_{i,j} = 1] = \mathbb{P}[\xi_{i,j} = -1] = \frac{1}{2}$$

for all $i \in [m], j \in [N]$. Further, let $s \in [m]$ be a specified *column sparsity* and let $\{(\delta_{i,j})_{i=1}^m\}_{j=1}^N$ be random vectors taking values in $\{0,1\}^m$. $\{(\delta_{i,j})_{i=1}^m\}_{j=1}^N$ are selector variables with the following properties:

- for $1 \leq j_1 < j_2 \leq N$, the vectors $(\delta_{i,j_1})_{i=1}^m$ and $(\delta_{i,j_2})_{i=1}^m$ are independent;
- for a fixed index $j \in [N]$, the entries of $(\delta_{i,j})_{i=1}^m$ are negatively correlated, *i.e*., for $1 \leq k \leq m$

$$\forall 1 \leq i_1 < i_1 \ldots < i_k \leq m, \;\; \mathbb{E}_{\delta_{i_1,j},\ldots,\delta_{i_k,j}}\left[\prod_{t=1}^k \delta_{i_t,j}\right] \leq \prod_{t=1}^k \mathbb{E}_{\delta_{i_t,j}}[\delta_{i_t,j}] = \left(\frac{s}{m}\right)^k; \tag{12}$$

- for a fixed index $j \in [N]$, the vector $(\delta_{i,j})_{i=1}^m$ has exactly s nonzero entries, *i.e*.,

$$\sum_{i=1}^m \delta_{i,j} = s.$$

The *sparse Johnson-Lindenstrauss transform (SJLT)* with column sparsity $s \in [m]$ is defined as

$$\mathbf{S}_{i,j} = \frac{1}{\sqrt{s}} \xi_{i,j} \delta_{i,j}.$$

This type of sketch matrix is particularly suitable of dimensionality reductions for large problem instances. If the column sparsity s is sufficiently small, then the matrix-vector product $\mathbf{Sx}$ can be computed very efficiently in time $O(s\|\mathbf{x}\|_0)$ where $\|\mathbf{x}\|_0 := |\{i \in [N] : \mathbf{x}_i \neq 0\}|$.

A simple way to sample a sparse Johnson-Lindenstrauss transform was proposed by Kane and Nelson [KN14]: for each column $j \in [N]$, they sample s row-indices $i_1, \ldots, i_s$ from $[m]$ without replacement and set

$$\delta_{i,j} = 1 \Leftrightarrow i \in \{i_1, \ldots, i_s\}.$$

It is clear that the vectors $(\delta_{i,j})_{i=1}^m, j = 1, \ldots, N$ are independent and each have exactly s non-zero entries. Further, we see that the entries are also negatively correlated. Let $j \in [N]$ and let $1 \leq i_1 < i_2 \leq m$. It holds

$$\mathbb{E}_{\delta_{i_1,j},\delta_{i_2,j}}[\delta_{i_1,j}\delta_{i_2,j}] = \mathbb{E}_{\delta_{i_1,j}}[\delta_{i_1,j}\mathbb{E}_{\delta_{i_2,j}}[\delta_{i_2,j} \mid \delta_{i_1,j} = 1]] = \frac{s}{m}\max\left\{0, \frac{s-1}{m-1}\right\} \leq \left(\frac{s}{m}\right)^2.$$

Now, property (12) follows by induction.

4.2 Main Result

The main result of Yang *et al*. is concerned with bounding the *true loss* on the given training examples. Let $\hat{f}$ be the sketched estimator. Then, the *true loss* on the training examples is defined as

$$\|\hat{f} - f^*\|_N^2 := \frac{1}{N}\sum_{n=1}^{N}(\hat{f}(\mathbf{x}_n) - f^*(\mathbf{x}_n))^2.$$

A theoretical bound on this quantity can be understood as a denoising guarantee. It says that we are able to infer the true target values from noisy measurements. However, this is not a classical performance guarantee from a machine learning point of view. In a typical learning scenario, we train our model on the training set and then want to apply it to new samples. A generalization guarantee would be a bound on the expected risk $R(\hat{f})$. The theorem employs two important concepts:

Kernel Complexity. Figure 1 illustrates the need for a regularization term if we want to learn by minimizing the empirical risk. The regularizer allows us to trade off a good fit to the (noisy) data and the smoothness of the estimator. In general, the amount of regularization that is required for a particular regression problem depends on several parameters: the number of training examples N, the noise level σ^2, and the *complexity* of the *hypothesis set* $\mathcal{H}$. *Rademacher complexities* are a common theoretical tool that allows us to quantify this intuition for complexity (see, *e.g*., [Kol11]).

Let $r > 0$ and let $\mathcal{F} := \{\mathcal{L}(f(\cdot),\cdot) : f \in \mathcal{H}, R(f) \leq r\}$ where $\mathcal{L}(f(\mathbf{x}),y) = (f(\mathbf{x}) - y)^2$ is the squared loss. Further, let $(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)$ be independent samples from the joint probability distribution P over $\mathcal{X}\times\mathbb{R}$. Then, the *(data dependent) Rademacher complexity* is defined as

$$R_N(\mathcal{F}) := \mathbb{E}_{\xi_1,...,\xi_N}\left[\sup_{f\in\mathcal{F}}\frac{1}{N}\left|\sum_{n=1}^{N}\xi_n f(\mathbf{x}_n,y_n)\right|\,\middle|\,(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\right]$$

where $\{\xi_n\}_{n=1}^N$ are independent Rademacher variables (also independent of the samples $(\mathbf{x}_n,y_n)$ for $n \in [N]$). It can be shown [Kol11, Propositions 3.2 and 3.4] that

$$\mathbb{E}\left[\sup_{f\in\mathcal{F}}\frac{1}{N}\left|\sum_{n=1}^{N}\xi_n f(\mathbf{x}_n,y_n)\right|\,\middle|\,(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)\right] \leq C\underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}\min(\mu_n,r^2)\right)^{1/2}}_{\mathcal{R}(r)} \tag{13}$$

where $\mu_1 \geq \ldots \geq \mu_N$ are the eigenvalues of the rescaled kernel matrix $\frac{1}{N}\mathbf{K}$. The right-hand side of (13) is called *(empirical) kernel complexity function*. It quantifies the effective complexity of $\mathcal{H}$ for a given set of training examples. Based on the kernel complexity function, Yang *et al*. introduce the following two quantities.

Definition 6 (Critical radius)**.** *The smallest $r_N > 0$ for which the inequality*

$$\frac{\mathcal{R}(r_N)}{r_N} \leq \frac{r_N}{\sigma}$$

holds is called the critical radius of $\mathbf{K}$.

Definition 7 (Statistical dimension)**.** *Let r_N be the critical radius. The* statistical dimension *is defined as*

$$d_N := \operatorname*{arg\,min}_{n=1,...,N}\{\mu_n \leq r_N^2\}.$$

It corresponds to the number of eigenvalues that are greater than the squared critical radius.

Example 3. *We want to investigate how the critical radius of the polynomial kernel $k(\mathbf{x},\mathbf{x}') = (\mathbf{x}\cdot\mathbf{x}')^D$ of degree $D \in \mathbb{N}$ on $\mathbb{R}^d$ scales with the number of data points N and the noise level σ^2. We saw in example 2*

that the polynomial kernel corresponds to the dot product in a $D_{\mathcal{H}} := \begin{pmatrix} D+d-1 \\ d \end{pmatrix}$ *dimensional feature space. Thus, the kernel matrix* $\mathbf{K}$ *has at most rank* $\min(N, D_{\mathcal{H}})$*. If* $N > D_{\mathcal{H}}$*, then we get*

$$\mathcal{R}(r) = \left(\frac{1}{N} \sum_{n=1}^{N} \min(\mu_n, r^2) \right)^{1/2} = \left(\frac{1}{N} \sum_{n=1}^{D_{\mathcal{H}}} \min(\mu_n, r^2) \right)^{1/2} \leq \left(\frac{D_{\mathcal{H}}}{N} r^2 \right)^{1/2}$$

and thus for $r^2 \geq r_N^2 = \sigma^2 \frac{D_{\mathcal{H}}}{N}$ *we have*

$$\mathcal{R}(r) \leq \left(\frac{D_{\mathcal{H}}}{N} r^2 \right)^{1/2} \leq \frac{r^2}{\sigma}.$$

K-satisfiability. A matrix $\mathbf{S}$ for which the property (10) in the Johnson-Lindenstrauss lemma holds is called an ε-*isometry*. The property says that pair-wise distances are approximately being preserved. Yang *et al*. called matrices for which a similar property for the size-reduction of kernel matrices holds $\mathbf{K}$-*satisfiable*. Let $\frac{1}{N}\mathbf{K} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ be an eigendecomposition of the rescaled kernel matrix $\frac{1}{N}\mathbf{K}$ where $\mathbf{U} \in \mathbb{R}^{N \times N}$ is orthogonal and $\mathbf{D} = \operatorname{diag}(\mu_1, ..., \mu_N)$ is a diagonal matrix with the entries being the eigenvalues of $\mathbf{K}$ in decreasing order. Let d_N be the statistical dimension of $\mathbf{K}$. We decompose $\mathbf{U}$ and $\mathbf{D}$ into blocks as

$$\mathbf{U} = [\mathbf{U}_1 \mid \mathbf{U}_2] \qquad \text{and} \qquad \mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & 0 \\ 0 & \mathbf{D}_2 \end{bmatrix}$$

where $\mathbf{U}_1 \in \mathbb{R}^{N \times d_N}, \mathbf{D}_1 = \mathbb{R}^{d_N \times d_N}$, and $\mathbf{U}_2 \in \mathbb{R}^{N \times (N-d_N)}, \mathbf{D}_2 \in \mathbb{R}^{(N-d_N) \times (N-d_N)}$ are the matrices corresponding to the d_N largest and the $(N - d_N)$ smallest eigenvalues of $\mathbf{K}$. Intuitively, a sketch matrix $\mathbf{S}$ is *good* if the dimensionality reduced matrix $\mathbf{S}\mathbf{U}_1$ is almost an isometry and the dimensionality reduced matrix $\mathbf{S}\mathbf{U}_2$ has a small operator norm (*i.e*., the relative importance of certain eigenvectors is not increased by projecting them onto the random linear subspace defined by $\mathbf{S}$). Formally, Yang *et al*. defined that a sketch matrix $\mathbf{S}$ is $\mathbf{K}$-*satisfiable* if

$$\|(\mathbf{S}\mathbf{U}_1)^T(\mathbf{S}\mathbf{U}_1) - \mathbf{I}\|_{l_2^{d_N} \mapsto l_2^{d_N}} \leq \frac{1}{2} \qquad \text{and} \qquad \|\mathbf{S}\mathbf{U}_2\mathbf{D}_2^{1/2}\|_{l_2^{N-d_N} \mapsto l_2^m} \leq c r_N$$

where r_N is the critical radius of $\mathbf{K}$ and c is a universal constant.

Remark 3 (Practical application)**.** *In practice, we do not explicitly check if a sampled sketch matrix* $\mathbf{S}$ *is* $\mathbf{K}$*-satisfiable. Checking this condition would require the computation of an eigendecomposition of* $\mathbf{K}$*. Computing such a decomposition takes times* $O(N^3)$ *and is thus infeasible for large* N*. If we apply random sketches for the dimensionality reduction of kernel matrices, then we rely on the fact that, for example, subgaussian sketch matrices are* $\mathbf{K}$*-satisfiable with high probability (See [YPW, Lemma 5]).*

Main result. We conclude this section with the main result of Yang *et al*. As the primary focus of this seminar paper lies in the possible applications of sketched kernel ridge regression in machine learning, we do not go into detail and direct the reader to [YPW] for further information.

Theorem 4 ([YPW, Theorem 2])**.** *Let* $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$ *be independent samples from the Gaussian noise model. Let* $\mathbf{S} \in \mathbb{R}^{m \times N}$ *be a* $\mathbf{K}$*-satisfiable sketch matrix. Then, for all regularization parameters* $\lambda > 2r_N^2$*, the minimum* $\hat{f}$ *of* $(P_{\mathbf{S},d})$ *satisfies the bound*

$$\|\hat{f} - f^*\|_N^2 \leq c_u(\lambda + r_N^2)$$

with probability at least $1 - c_1 e^{-c_2 N r_N^2}$.

5 Evaluation

In this section, we present the results of our experimental evaluation. While Yang *et al*. performed several numerical simulations in order to show the validity of their derived bound, they did not perform any experiments from a machine learning point of view. In order to evaluate the performance of the sketched KRR estimator in a machine learning application, we conducted several experiments.

For all of our experiments, we use the *Gaussian kernel*

$$k(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{1}{2h}\|\mathbf{x}-\mathbf{x}'\|_2^2\right)$$

with *bandwidth* $h > 0$, which is a positive definite kernel on $\mathbb{R}^D$.

Whenever we want to learn an estimator using kernel ridge regression with a Gaussian kernel, we need to fix the bandwidth h and the regularization parameter λ. While theorem 4 gives us a principled way of choosing λ, the minimizer of $(P_{\mathbf{S},d})$ is empirically very sensitive to the choice of λ. Therefore, we usually determine all parameters numerical for optimal performance.

A common way of determining these parameters in machine learning is *validation*. First note that we cannot simply optimize over λ and h on our training examples $(\mathbf{x}_1,y_1),...,(\mathbf{x}_N,y_N)$. This would result in the trivial choice $\lambda = h = 0$. For this reason, we use a dedicated *evaluation set* of i.i.d. samples $(\tilde{\mathbf{x}}_1,\tilde{y}_1),...,(\tilde{\mathbf{x}}_K,\tilde{y}_K)$. We use these samples in order to evaluate the performance of a learned estimator. To this end, we fix a finite set of possible values for each parameter and then learn one estimator for each parameter choice. We finally select the estimator that achieves minimum empirical risk on the validation set. For our experiments, we chose the possible values $\{10^{-9},10^{-8},10^{-7},...,10^{2},10^{3}\}$ for both parameters.

Sketching vs bootstrap sampling. It is clear that we get an improved runtime by reducing the size of the kernel matrix. However, there are two possible ways to reduce its size: sketching and *bootstrap sampling*. The idea of bootstrap sampling is to generate a new (potentially smaller) dataset by sampling from the original dataset. Thus, by sampling only a small number of points, we can also reduce the size of the kernel matrix. In this experiment, we compare the performance of the sketched estimator to that of a bootstrapped sampled estimator.

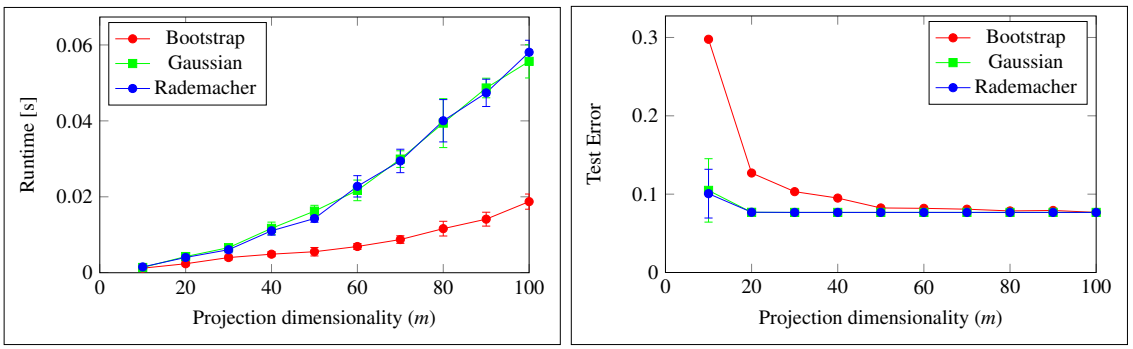
We generate a synthetic dataset of $N = 1200$ examples $\{(\mathbf{x}_n,y_n)\}_{n=1}^N$. The inputs $\mathbf{x}_n$ are sampled i.i.d. from a uniform distribution over the interval $[1,-1]$ and the targets follow the Gaussian noise model

$$y_n = f^*(\mathbf{x}_n) + \omega_n = \sin(2\pi\mathbf{x}_n) + \omega_n$$

for $n \in [N]$ where $\omega_1,...,\omega_N \overset{iid}{\sim} \mathcal{N}(0,0.25)$. We divide the dataset into 100 training examples, 1000 testing examples, and 100 evaluation examples. For the comparison, we use Gaussian and Rademacher sketch matrices. Figure 2 reports the test error as a function of the size of the kernel matrix. The results show clearly that random sketches perform significantly better than bootstrap sampling while still maintaining a reasonable runtime. At a projection dimensionality of 20, the sketched regressor performs on par with the regressor that is trained on the full dataset.

The results of this experiment can be seen as a proof of concept. They show that random sketches outperform a widely used baseline method for speeding up training.

Sparsity of the sparse JL transform. In this experiment, we evaluate the performance of the sketched estimator when using the sparse Johnson-Lindenstrauss transform. We use the implementation proposed by Kane and Nelson (See section 4.1). We are interested in the test error as a function of the sparsity. To this end, we use the same dataset as in the previous experiment at a fixed projection dimensionality of $m = 30$ and vary the sparsity. We found that for this particular dataset the sparsity can be set arbitrarily low (*i.e.*, $s = 1$). However, we attribute this to the the high number of training examples relative to the dimensionality of the data and the low noise level. For more conclusive results, it is important to investigate this relationship further on real-world data.



(a) (b)

Figure 2: (a) The mean runtime and standard deviation as a function of the size of the kernel matrix measured on a quad-core machine 2.8GHz; (b) the mean test error and standard deviation as a function of the size of the kernel matrix. Each measurement is an average over 10 runs.

Dataset	**Size**	**Features**	**#Training**	**#Test**	**#Evaluation**
COMPRESSIVE [com]	1,030	8	515	309	206
AIRFOIL [air]	1503	5	752	451	300
MPG [mpg]	392	7	196	118	78
YACHT [yac]	308	6	154	93	61

Table 1: The table lists all datasets that we evaluate on. The splits into training, test, and evaluation sets are obtained by randomly splitting (50%, 30%, 20%).

Performance on real-world data. In this experiment, we evaluate the performance of the sketched KRR estimator on real-world data. For this purpose, we chose four datasets from the UCI machine learning repository [Lic13]. Table 1 summarizes the chosen dataset. We split each set in 50% training set, 30% test set, and 20% evaluation set. Figure 3 reports the test errors as functions of the sizes of the corresponding kernel matrices. The results are similar to the ones obtained on synthetic data and the performance of the sketched KRR estimator compares favorably to the one of the unsketched KRR estimator.

Sparsity of the sparse JL transform. Based on the results that we obtained in the previous experiment, we now investigate the performance of the sketched KRR estimator when using the sparse Johnson-Lindenstrauss transform. To this end, we fix a projection dimensionality m for each dataset (COMPRESSIVE: $m = 300$; AIRFOIL: $m = 225$; MPG: $m = 125$; YACHT: $m = 75$) and vary the column sparsity s. Figure 4 reports the test error as a function of the column sparsity s. There are two significant observations that we can make: Firstly, the results show that we potentially can achieve outstanding performance even for very low sparsity values. Secondly, we see that the performance can vary considerably. This indicates that the probability of obtaining a $\mathbf{K}$-satisfiable sketch matrix decreases with decreasing values of the column sparsity s.

The sparse Johnson-Lindenstrauss transform is of particular interest in practice because it allows for a very efficient computation of the matrix vector product $\mathbf{Sx}$ in time $O(s\|\mathbf{x}\|_0)$. In our experiments, we measured the computation time as a function of the column sparsity. We implemented the sparse Johnson-Lindenstrauss transform using dedicated sparse data structures for maximal performance benefits. Unfortunately, we found the difference in computation time between a full Rademacher sketch matrix and a sparse Johnson-Lindenstrauss transform to be insignificant ($< 1\%$ reduction in processing time) for our data sets. However, this does not imply that the sparse Johnson-Lindenstrauss transform cannot reduce processing time significantly in general. For our problem sizes ($N \approx 500$), the matrix vector product $\mathbf{SK}$ can be computed quickly, regardless of the choice of $\mathbf{S}$. We suppose that the sparse Johnson-Lindenstrauss transform can reduce computation time significantly for very large problem sizes. Further experimental investiga-

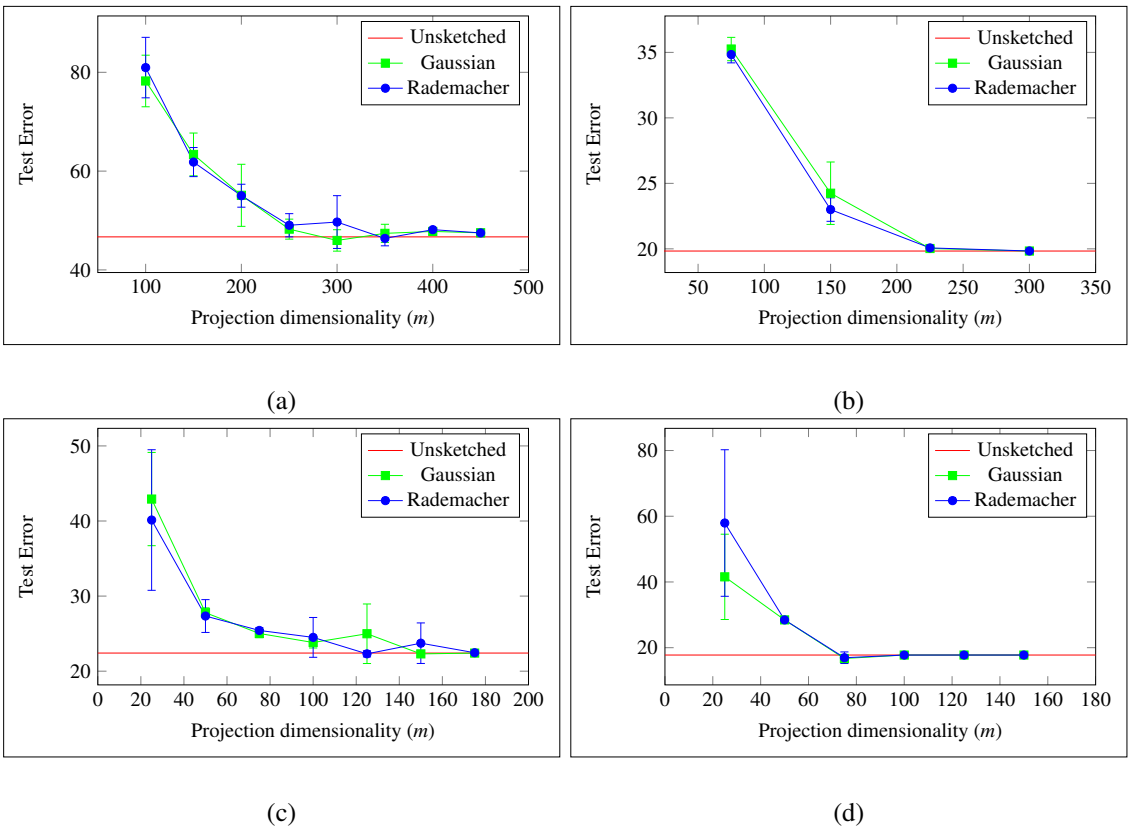


Figure 3: The plots show the mean test errors and standard deviations as functions of the projection dimensionality. Each measurement is an average over 5 runs. (a) COMPRESSIVE; (b) AIRFOIL; (c) MPG; (d) YACHT.

tion needs to be carried out in order to quantify the reduction in processing time for large scale real-world applications.

Remark 4 (Performance)**.** *The particular performance levels* (i.e. *test errors) that we obtained are not necessarily comparable to results obtained by other authors on these datasets. By employing a finer grid search, the individual performance scores can most definitely be improved. However, the goal of these experiments was to evaluate the performance of the sketched kernel ridge regression approach* relative *to the unsketched approach. To this end, a somewhat coarse grid search does not skew the results.*

6 Conclusion

In this seminar report, we discussed a recent approach proposed by Yang *et al*. that aims at improving the time that is required in order to learn a kernel ridge regressor [YPW]. They proposed to reduce the size of the corresponding optimization problem using random sketches. They showed that we obtain a performance guarantee for the sketched estimator in form of a bounded true loss on the training examples. We gave a thorough introduction to kernel methods. In this introduction, we discussed positive definite kernels and introduced the reproducing kernel Hilbert space (RKHS). Based on the RKHS we stated the representer theorem that allowed us to learn an estimator over the (possibly infinite dimensional) RKHS. Then, we gave a brief introduction to random sketches and summarized main theorem of Yang *et al*. Finally, we reported the findings of our numerical experiments. We found that the sketched estimators perform on par with the unsketched estimators even for sketch matrices $\mathbf{S} \in \mathbb{R}^{m \times N}$ with $m \ll N$. These observations could be made on synthetic data as well as on real-world data. This presents sketched kernel ridge regression as

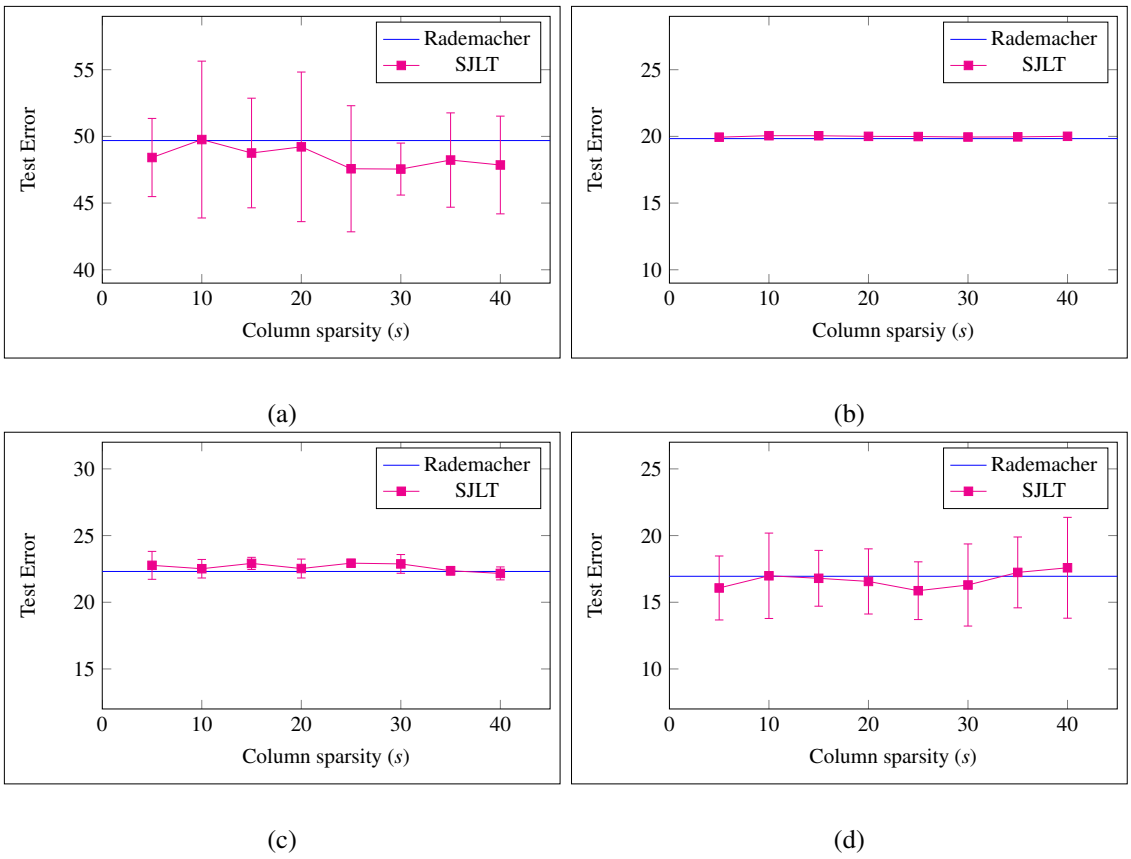


Figure 4: The plots show the mean test errors and standard deviations as functions of the column sparsity. Each measurement is an average over 25 runs. (a) COMPRESSIVE; (b) AIRFOIL; (c) MPG; (d) YACHT.

practically relevant tool for large problem instances (*i.e.* the number of data points N is large). Furthermore, we found that even for small values of the column sparsity s, the sparse Johnson-Lindenstrauss transform is a viable alternative to traditional Rademacher sketch matrices. While the sparsity did not have a significant impact on the computation time for our problem instances ($N \approx 500$), we suppose that the sparse Johnson-Lindenstrauss transform can reduce processing time significantly for very large problem instances. We leave the experimental investigation of this claim as future work.

References

[air] Airfoil self-noise dataset. http://archive.ics.uci.edu/ml/datasets/Airfoil+Self-Noise. Accessed: 2015-06-22.

[BDN15] J. Bourgain, S. Dirksen, and J. Nelson. Toward a unified theory of sparse dimensionality reduction in euclidean space. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC '15, pages 499–508, New York, NY, USA, 2015. ACM.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[com] Concrete compressive strength dataset. http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength. Accessed: 2015-06-22.

[FR13] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[GVL13] G. H. Golub and C. F. Van Loan. *Matrix Computations (4th Edition)*. Johns Hopkins University Press, Baltimore, MD, USA, 2013.

[JL84] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.

[KN14] D. M. Kane and J. Nelson. Sparser Johnson-Lindenstrauss transforms. *J. ACM*, 61(1):4:1–4:23, January 2014.

[Kol11] V. Koltchinskii. *Oracle Inequalities in Empirical Risk Minimization and Sparse Recovery Problems*. Springer-Verlag, Heidelberg, 2011.

[Lic13] M. Lichman. UCI machine learning repository, 2013.

[mpg] Auto mpg dataset. http://archive.ics.uci.edu/ml/datasets/Auto+MPG. Accessed: 2015-06-22.

[Mur12] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[SGV98] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 515–521, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[SS01] B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

[yac] Yacht hydrodynamics dataset. http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics. Accessed: 2015-06-22.

[YPW] Y. Yang, M. Pilanci, and M. J. Wainwright. Randomized sketches for kernels: Fast and optimal non-parametric regression.